台科大 111 學年度「彩色影像處理」 作業一：空間濾波(不得使用現成影像濾波函式)

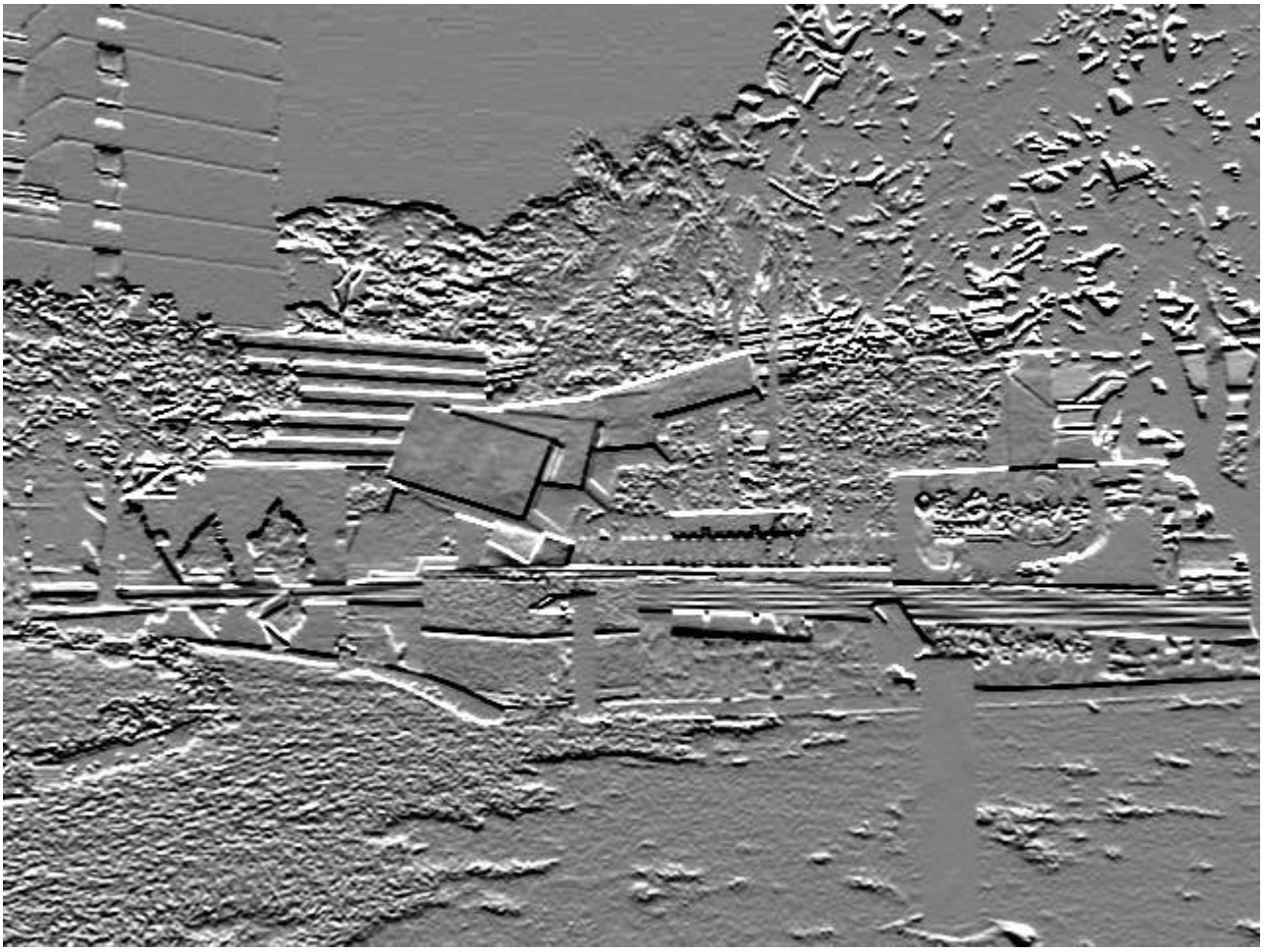
M11125016 蕭強

Programming language: Python

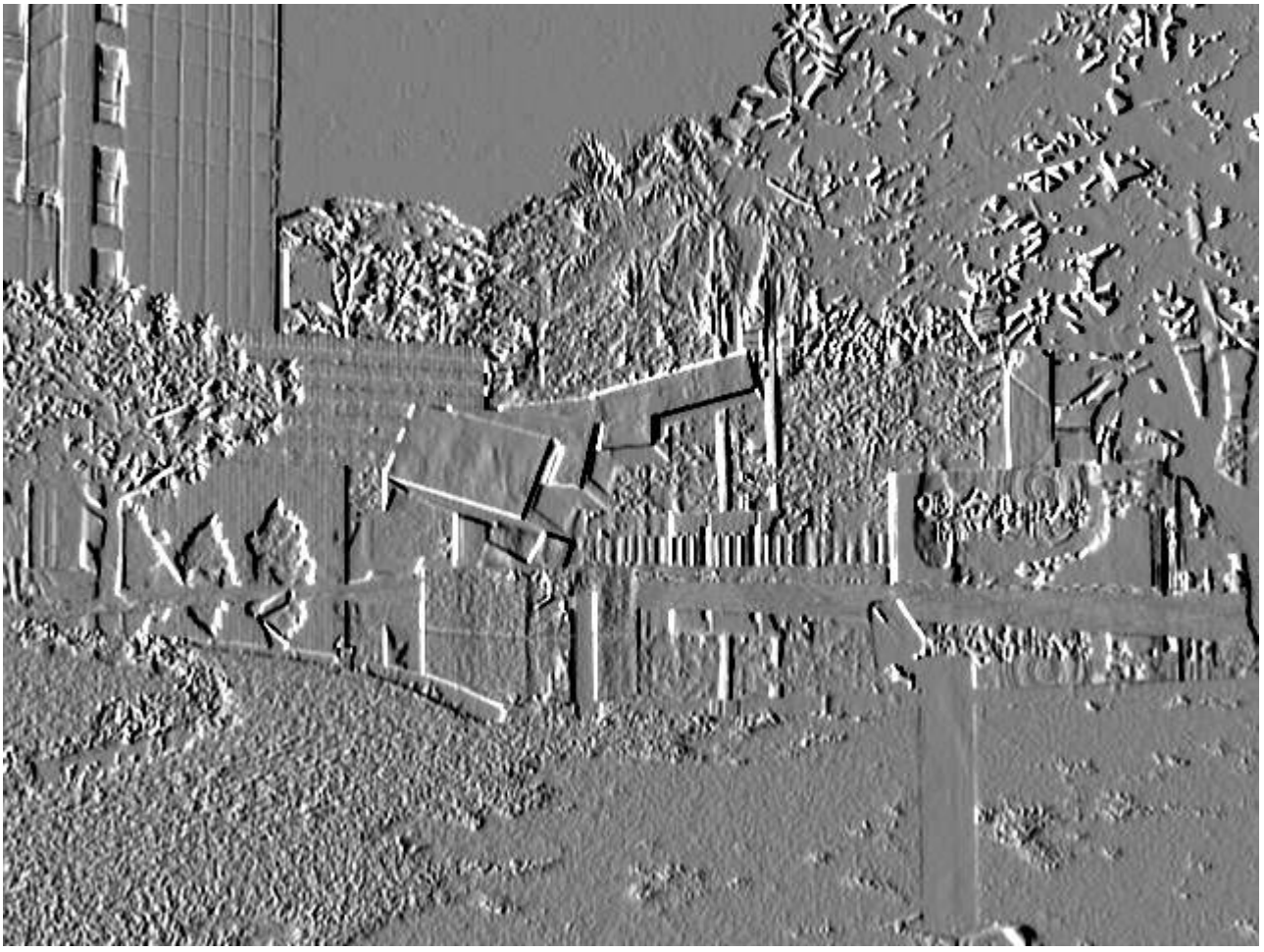
I. 第一題「影像邊緣偵測」:

```
import numpy as np
import cv2

def processingImg(imagePath):
    image = cv2.imread(imagePath, 0)
    # 4. 將影像轉換成 double 格式，數值範圍在[0 1]之間。
    image = image / 255  # float64
    return image

def edgeDetect(image, kernel):
    # 4. 用雙層迴圈由左而右，由上而下讀取以(x,y)為中心的 3×3 影像區域。
    # 5. 將 3×3 影像區域點對點乘上圖 1 Sobel 濾鏡數值矩陣後，將數值總和存入輸出影像的(x,y) 位置。
    img_row, img_col = image.shape
    ker_row, ker_col = kernel.shape
    pad = int((ker_col - 1) / 2)
    image = np.pad(image, (pad, pad), 'symmetric')
    output = np.zeros((img_row, img_col), dtype=np.float64)
    for y in range(img_row):
        for x in range(img_col):
            cim = image[y:y + 3, x:x + 3]
            result = (cim * kernel).sum()
            output[y, x] = result
    return output

def embossing(image):
    # 6. 將濾波後的影像加上 0.5，呈現近似圖 2(b)的浮雕影像。
    image = image + 0.5
    return image

def threshold(image, threshold):
    # 7. 分別將濾波後的影像開絕對值，再二值化(門檻值自訂)，用 bitor (bitwise or)或直接相加，產生近似圖 2(c)的輪廓影像。
    # image = abs(image)
    h, w = image.shape
    img_thres = np.zeros((h, w))
    for y in range(0, h):
```

```
        for x in range(0, w):
            pixel = image[y, x]
            if pixel < threshold:
                np_pix = 0
            else:
                np_pix = 1
            img_thres[y, x] = np_pix
    return img_thres

# filters
sobel_hor = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
sobel_ver = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

# load image
raw_img = processingImg('ntust_gray.jpg')

# 4. 用雙層迴圈由左而右，由上而下讀取以(x,y)為中心的 3×3 影像區域。
# 5. 將 3×3 影像區域點對點乘上圖 1 Sobel 濾鏡數值矩陣後，將數值總和存入輸出影像的(x,y) 位置。
sobel_hor_img = edgeDetect(raw_img, sobel_hor)
sobel_ver_img = edgeDetect(raw_img, sobel_ver)

# 6. 將濾波後的影像加上 0.5，呈現近似圖 2(b)的浮雕影像。
sobel_hor_emb_img = embossing(sobel_hor_img)
sobel_ver_emb_img = embossing(sobel_ver_img)

# 7. 分別將濾波後的影像開絕對值，再二值化(門檻值自訂)，用 bitor (bitwise or)或直接相加，產生近似圖 2(c)的輪廓影像。
sobel_hor_thres_img = threshold(sobel_hor_img, 0.3)
sobel_ver_thres_img = threshold(sobel_ver_img, 0.3)
doubel_sobel_img = sobel_hor_thres_img + sobel_ver_thres_img

sobel_hor_emb_img = sobel_hor_emb_img * 255
sobel_ver_emb_img =sobel_ver_emb_img * 255
doubel_sobel_img = doubel_sobel_img * 255
cv2.imwrite("sobel_hor_emb_img.jpg", sobel_hor_emb_img)
cv2.imwrite("sobel_ver_emb_img.jpg", sobel_ver_emb_img)
cv2.imwrite("doubel_sobel_img.jpg", doubel_sobel_img)
```



圖表 1 水平濾波浮雕處理



圖表 2 垂直濾波浮雕處理

圖表 3 雙向濾波輪廓處理

II. 第二題「Unsharp Masking(USM)影像銳化」

```python
import numpy as np
import cv2

def processingImg(imagePath):
    image = cv2.imread(imagePath, 0)
    # 4. 將影像轉換成 double 格式，數值範圍在[0 1]之間。
    image = image/255  # float64
    return image

def unsharpMask(image, kernel):
    # 5. 用雙層迴圈對 n x n 濾鏡(均值濾鏡或高斯濾鏡)做影像模糊化，獲得近似圖 3(b)的結果。
    img_row, img_col = image.shape
    ker_row, ker_col = kernel.shape
    pad = int((ker_col - 1) / 2)
    image = np.pad(image, (pad, pad), 'symmetric')
    output = np.zeros((img_row, img_col), dtype=np.float64)
    for y in range(img_row):
        for x in range(img_col):
            cim = image[y:y + 3, x:x + 3]
            result = (cim * kernel).sum()
            output[y, x] = result
    return output

# filters
mean_filter = np.ones((3,3))/9
gaussian_Filter = np.array([[1, 2, 1], [2, 4, 2], [1, 2, 1]]) / 16

# load image
raw_img = processingImg('ntust_gray.jpg')
# 5. 用雙層迴圈對 n x n 濾鏡(均值濾鏡或高斯濾鏡)做影像模糊化，獲得近似圖 3(b)的結果。
blur_image = unsharpMask(raw_img, gaussian_Filter)
# 6. 利用原圖與模糊影像的差異，加上原圖，獲得類似圖 3(c)的銳利影像。
usm_image = 0.8 * (raw_img - blur_image) + raw_img  # 0.8(raw - processed) + raw

raw_img = raw_img * 255
blur_image = blur_image * 255
usm_image = usm_image * 255

cv2.imwrite("gaussian_Filter_image.jpg", blur_image)
cv2.imwrite("gaussian_usm_image.jpg", usm_image)
```

圖表 4 均值濾波處理

圖表 5 高斯濾波處理

圖表 6 均值濾波銳利化

圖表 7 高斯濾波銳利化